KE LAB
Assignment 3
Data Cleaning And Filling Missing Values

1. Codes for missing values:

Missing value is represented by * in database.

DATABASE:

T1 a b * e
T2 b d
T3 a c * f
T4 * d f
T5 c d e

Method 1:

By ignoring the missing values.

Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    ifstream f;
    f.open("db.txt");
    string s;

    ofstream fout;
    fout.open("ignoring_tuple_output.txt");
    while(getline(f, s)){
        string t = s.substr(0, 2);
        fout<<t<<" ";
        for(int i=2;i<s.size();i++){
            if(isalpha(s[i]) ){
                fout<<s[i]<<" ";
            }
        }
        fout<<endl;
    }
    f.close();
    fout.close();
    return 0;
}
```

OUTPUT:

T1 a b e
T2 b d
T3 a c f
T4 d f
T5 c d e

Method 2:

Filling the value by a CONST.

Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    ifstream f;
    f.open("db.txt");
    string s;

    const string STR = "a";
    ofstream fout;
    fout.open("replace_with_constant_output.txt");
    while(getline(f, s)){
        string t = s.substr(0, 2);
        fout<<t;
        for(int i=2;i<s.size();i++){
            if(s[i] == '*'){
                fout<<STR;
            }
            else
                fout<<s[i];
        }
        fout<<endl;
    }
    f.close();
    fout.close();
    return 0;
}
```

OUTPUT:

T1 a b a e
T2 b d

T3 a c a f
T4 a d f
T5 c d e

Method 3:
Replacing the missing value by most frequent value.
Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    ifstream f;
    f.open("db.txt");
    string s;
    map<char, vector<string> >m;
    while(getline(f, s)){
        string t = s.substr(0, 2);
        for(int i=2;i<s.size();i++){
            if(isalpha(s[i])){
                m[s[i]].push_back(t);
            }
        }
    }
    f.close();

    map<char, vector<string> >::iterator i;
    int mi = -1;
    string STR;
    for(i = m.begin();i!=m.end();i++){
    //  cout<<i->first<<" ";
    //  cout<<i->second.size()<<endl;
        if(int(i->second.size()) > mi){
            mi = i->second.size();
            STR = i->first;
        }
    }
    //cout<<STR<<" "<<mi;
    f.open("db.txt");
    ofstream fout;
    fout.open("replace_with_mode_output.txt");
    while(getline(f, s)){
        string t = s.substr(0, 2);
        fout<<t;
        for(int i=2;i<s.size();i++){
            if(s[i] == '*'){
                fout<<STR;
            }
            else
                fout<<s[i];
        }
        fout<<endl;
    }
    f.close();
    fout.close();
    return 0;
}
```

OUTPUT:

T1 a b d e
T2 b d
T3 a c d f
T4 d d f
T5 c d e

2. Codes for removing noise:

DATABASE:

T1 4 8 15 21 21 24 25 28 34
T2 5 6 8 12 14 16 24 27 29
T3 7 9 12 18 21 24 30 32 37

Method 1:
Replacing each value by mean of that bin(of size = 3)
Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    ifstream f;
    f.open("vs.txt");
    string s;
    map<string, vector<int> >trans;
    map<string, vector<int> >::iterator t;
    while(getline(f, s)){
        string t = s.substr(0, 2);
        for(int i=2;i<s.size();i++){
            if(s[i] != ' '){
                vector<char>st;
                while(s[i] != ' ' && i<s.size()){
                    st.push_back(s[i]);
                    i++;
                }
                int sum = 0;
                reverse(st.begin(), st.end());
                for(int i=0;i<st.size();i++){
                    int j = int(st[i] - '0');
                    sum += j*pow(10, i);
                }
                trans[t].push_back(sum);
                //cout<<sum<<" ";
            }
        }
        //cout<<endl;
    }
    f.close();

    ofstream fout;
    fout.open("replace_with_mean_output.txt");
    vector<int>mean(trans.size());
    for(t = trans.begin();t != trans.end();t++){
        fout<<t->first<<" ";
        int n = t->second.size();
        const int DIVIDE = 3;
        for(int i=0;i<n;i++){
            int m = 0;
            for(int j=i;j<i+DIVIDE;j++)
                m += t->second[i];
            m = m/DIVIDE;
            for(int j=i;j<i+DIVIDE;j++){
                fout<<m<<" ";
            }
            i += DIVIDE;
        }
        fout<<endl;
    }
    fout.close();
    return 0;
}
```

OUTPUT:

T1 4 4 4 21 21 21 34 34 34

T2 5 5 5 14 14 14 29 29 29

T3 7 7 7 21 21 21 37 37 37

Method 2:

Replacing each value with near boundary

Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    ifstream f;
    f.open("vs.txt");
    string s;
    map<string, vector<int> >trans;
    map<string, vector<int> >::iterator t;
    while(getline(f, s)){
        string t = s.substr(0, 2);
        for(int i=2;i<s.size();i++){
            if(s[i] != ' '){
                vector<char>st;
                while(s[i] != ' ' && i<s.size()){
                    st.push_back(s[i]);
                    i++;
                }
                int sum = 0;
                reverse(st.begin(), st.end());
                for(int i=0;i<st.size();i++){
                    int j = int(st[i] - '0');
                    sum += j*pow(10, i);
                }
                trans[t].push_back(sum);
            }
        }
    }
    f.close();

    ofstream fout;
    fout.open("replace_with_boundary_output.txt");
    for(t = trans.begin();t != trans.end();t++){
        fout<<t->first<<" ";
        sort(t->second.begin(), t->second.end());
        int n = t->second.size();
        const int DIVIDE = 3;
        int i = 0;
        while(i<n){
            int l = t->second[i];
            int r = t->second[i + DIVIDE - 1];
            for(int j=i;j<i + DIVIDE;j++){
                if( (t->second[j] - l) <= (r - t->second[j]) )
                    fout<<l<<" ";
                else
                    fout<<r<<" ";
            }
            i += DIVIDE;
        }
        fout<<endl;
    }
    fout.close();
    return 0;
}
```

OUTPUT:

```
T1 4 4 15 21 21 24 25 25 34
T2 5 5 8 12 12 16 24 29 29
T3 7 7 12 18 18 24 30 30 37
```